MicroC/OS-II

James

Overview

- Not freeware
- Portable
- Small footprint
- Scalable
- Pre-emptive multitasking
- Real-time

Ports

- Mainly written in C
- Has some assembly for target-specific code
- Support many processors and boards:
  - ARM
  - Atmel
  - Freescale/Motorola
  - Fujitsu
  - IBM
  - Intel
  - Microchip
  - Misubishi
  - NEC
  - TI
  - etc.

OS Footprint

- It’s only an OS kernel, nothing else
- Less than 10k lines
- 6~24 Kbytes
  - Depending on applications

Multi-Tasking

- MicroC/OS-II supports multiple tasks
  - Up to 64 tasks
  - Up to 8 system tasks
  - Each task has a unique priority
- Task stack
  - Each task has its own stack
  - Stack size can be different
- Task states:
  - Dormant
  - Ready
  - Running
  - Waiting
  - Interrupt
- User tasks must be created first
- Each task is an infinite loop
- Shared resources are protected by semaphores

Scheduling and interrupts

- MicroC/OS-II is fully preemptive
- Always runs the highest priority task that is ready
- Tasks can be preempted by interrupts at any time
- Support nested interrupts
  - Up to 256 levels
- Interrupt handler will use the stack of the interrupted task

Implementation

- Initialization and startup
  - OSInit()
    - Used to initialize the internals of the OS
      - Ready list
      - TCB list
      - Message queue
      - OS event list
    - Must be called prior to create any object
    - Must be called before OSStart
    - Create an idle task
      - lowest priority
      - Cannot be deleted
      - Can be used to implement power management
    - Call port specific initialization code
  - OSStart()
    - Must have task(s) created before run
    - Find the next highest priority number
    - Move pointer to that task which is ready to run
    - Start that task

Implementation (2)

- Scheduling
  - OS_Sched():
    - Determine if a new, high priority task is ready to run
    - Allocate storage for CPU status register
    - Check if all (nested) ISRs have be done
    - Context switch
  - OS_SchedNew():
    - Find a new task to run
    - Look into the ready list
    - Get the highest priority task's priority number
  - OS_TASK_SW():
    - Start context switch
    - Trigger software interrupt
    - Call context switch handler OSCtxSw()

Implementation (3)

- Task related
  - OSStartHighRdy()
    - Load the context of the task
    - Execute the task
  - OSTaskChangePrio()
  - OSTaskCreate()
  - OSTaskDel()
  - etc.

Other OS services

- Mailbox
  - For data exchange between tasks
  - A task that reads an empty mailbox is blocked
  - Hold only one message in the mailbox
- Queue
  - For data exchanges between tasks
  - Hold system-wide messages
- Fix-size memory partition
- Time-related functions

OS extensions and tools

- µC/TCP-IP
- µC/USB Host
- µC/USB Devices
- µC/FS
- µC/Gui
- µC/Probe
- etc.

That’s it!
Questions?